# 𝔘Boost: Boosting with the Universum

Chunhua Shen, *Member, IEEE*, Peng Wang, Fumin Shen, and Hanzi Wang, *Senior Member, IEEE*

**Abstract**—It has been shown that the Universum data, which do not belong to either class of the classification problem of interest, may contain useful prior domain knowledge for training a classifier [1], [2]. In this work, we design a novel boosting algorithm that takes advantage of the available Universum data, hence the name 𝔘Boost. 𝔘Boost is a boosting implementation of Vapnik's alternative capacity concept to the large margin approach. In addition to the standard regularization term, 𝔘Boost also controls the learned model's capacity by maximizing the number of observed contradictions. Our experiments demonstrate that 𝔘Boost can deliver improved classification accuracy over standard boosting algorithms that use labeled data alone.

**Index Terms**—Universum, kernel methods, boosting, column generation, convex optimization.

## 1 Introduction

Universum inference means training a classifier with the help of Universum examples—the examples that do not belong to either of the classes of interest.

Suppose that, apart from the labeled training examples, we are given a set of unlabeled examples, termed Universum examples, which are collected from the same domain with the labeled examples and we know that these unlabeled data do not belong to either class. Now let us assume that all possible decision functions are categorized into a finite number of equivalence classes $\Gamma_1, \dots, \Gamma_l$. Functions in the same equivalence classes have the same training error, namely the empirical risk. Based on *the maximal contradiction on Universum principle* introduced by Vapnik and his colleagues [1], [2], here our goal is to find an equivalence class which has a large number of contradictions for training boosting classifiers. The contradiction happens when two functions in the same equivalence class have different signed outputs on a sample from the Universum.

Weston *et al.* [2] proposed an algorithm, termed Universum support vector machines (𝔘SVM), which has a regularization term for the Universum in addition to the standard SVM objective function. Their experimental results show that 𝔘SVM outperforms those SVMs without considering Universum data, *e.g.*, the standard SVM algorithm. Sinz *et al.* [3] analyzed the behavior of 𝔘SVM and show that 𝔘SVM makes the normal of the decision plane orthogonal to the principal directions of the Universum data. Then they present a least squares version of the 𝔘SVM algorithm, which has a closed-form solution, and discuss the relationship with Fisher discriminant analysis (FDA) and oriented principal component analysis (OPCA). Zhang *et al.* [4] proposed a graph based semi-supervised algorithm, which learns from the labeled data, unlabeled data and the Universum data simultaneously. Besides experiments on standard benchmark data sets, there are also some computer vision applications that utilize Universum data, such as human pose recognition [5], [6] and gender classification [7]. Note that as pointed out in [2], Universum training data should contain the information about the domain of the learning problem of interest. An example is handwritten digit recognition (see the experiments in Section 3). In this case, the Universum data can be handwritten symbols other than the digits to be classified from the same data set.

Inspired by the success of 𝔘SVM, in this work, we propose an boosting algorithm, referred to as 𝔘Boost. 𝔘Boost learns a strong classifier, with a minimal classification error on labeled data and a maximal contradiction on the Universum data. Given the optimization problem, we derive a *meaningful* Lagrange dual formulation. Based on the derived dual problem, we are able to iteratively solve the original optimization problem using the column generation technique from convex optimization [8]. Therefore, compared with standard boosting algorithms such as AdaBoost [9], the proposed 𝔘Boost has the following compelling properties.

- Besides labeled data from two classes, 𝔘Boost exploits Universum data as well. Improved classification accuracy is expected over conventional boosting algorithms that do not use Universum data.
  To our knowledge, this is the first boosting implementation of Vapnik's maximal contradiction on Universum principle.
- Inspired by the work of [10], we use column generation to facilitate the optimization of the formulated 𝔘Boost problem. So the proposed 𝔘Boost is totally corrective in the sense that at each iteration, the coefficients of all the selected weak classifiers are updated. In contrast, stage-wise boosting like AdaBoost only updates the selected weak classifier's coefficient at current iteration, usually leading to slower convergence.

Our experiments verify the usefulness of Universum data for classification problems, which confirms the conclusion on 𝔘SVM [2]. Before proceeding, we introduce some notation used in the sequel.

Boosting algorithms have been extensively studied in the literature [9], [10], [11], [12], [13], [14]. Boosting refers to a method for learning an accurate classifier by linearly combining a set of only moderately accurate weak classifiers. Similar to SVM, standard boosting is often trained on labeled data by minimizing a convex surrogate of the non-convex Bayes zero-one loss. Typically, the exponential loss (as in AdaBoost), logistic loss (LogitBoost) and hinge loss (LPBoost [15]) are employed. It has been shown in [16] that stage-wise boosting can be viewed as coordinate descent optimization in the functional space. Recently, Shen and Li [10] demonstrated that AdaBoost, LogitBoost and boosting with generalized hinge loss can all be seen as entropy maximization in the dual. They explicitly established the dual problems of a class of

- *C. Shen is with the Australian Center for Visual Technologies, and School of Computer Science at the University of Adelaide, SA 5005, Australia (e-mail: chunhua.shen@adelaide.edu.au).*
- *P. Wang is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. His contribution was made while visiting NICTA, Canberra Research Laboratory, Australia.*
- *F. Shen is with the School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China. His contribution was made while visiting School of Computer Science, University of Adelaide (e-mail: fumin.shen@gmail.com).*
- *H. Wang is with the School of Information Science and Technology, Xiamen University, Fujian 361005, China; Fujian Key Lab of the Brain-like Intelligent Systems (Xiamen University), Fujian 361005, China (e-mail: hanzi.wang@ieee.org); and School of Computer Science at the University of Adelaide, SA 5005, Australia. H. Wang was supported by NSFC under project 61170179 and by the Xiamen Science and Technology Planning project (3502Z20116005) of China.*

boosting algorithms. We follow this line of research in the sense that we also explicitly formulate the optimization problem using the $\ell_1$ norm regularization and derive the Lagrange dual problem. Based on the dual problem, we design a new boosting algorithm using column generation. As mentioned, the main difference is that the proposed $\mathfrak{U}$Boost considers the Universum data and therefore the optimization problem does not fall into the framework of [10].

The remaining content is organized as follows. In Section 2, we discuss the proposed $\mathfrak{U}$Boost algorithm. In Section 3, we show experiments of $\mathfrak{U}$Boost on various data sets. We demonstrate that the Universum data indeed help improve the test accuracy in general. We conclude this work in Section 4.

**Notation** The following notation is used throughout this paper. Suppose that we are given a set of $M$ *labeled* examples $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_M, y_M)\} \in \mathbb{R}^d \times \{1, -1\}$, and a set $\mathfrak{U} = \{\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_N\} \in \mathbb{R}^d$ called the Universum, which contains $N$ *unlabeled* examples. $\mathcal{H}$ is the entire set of possible weak classifiers, *i.e.*, $\mathcal{H} = \{h_k(\cdot) : \boldsymbol{x} \to \{1, -1\}, k = 1, \dots\}$. Note that the dimension of $\mathcal{H}$ can be infinite. Boosting algorithms learn a strong classifier $F(\boldsymbol{x}) = \sum_k w_k h_k(\boldsymbol{x})$, where $\boldsymbol{w} \geq \boldsymbol{0}$ is the coefficients of weak classifiers. We define the matrix $H \in \mathbb{Z}^{M \times K}$ such that it stores the predictions of all weak classifiers over labeled examples; *i.e.*, $H_{ik} = h_k(\boldsymbol{x}_i)$. Likewise, the matrix $H' \in \mathbb{Z}^{N \times K}$ is defined such that its $(j,k)$-th entry is $h_k(\boldsymbol{x}'_j)$. We use $H_i = [H_{i1}\ H_{i2}\ \cdots\ H_{ik} \cdots]$ to denote the $i$-th row of $H$, which is the output vector of all weak classifiers on example $\boldsymbol{x}_i$. Analogously, $H'_j$ denotes the $j$-th row of $H'$.

## 2 THE $\mathfrak{U}$BOOST ALGORITHM

We present the main results in this section. As explained previously, the main idea here is to exploit the unlabeled Universum data that do not belong to either class of the training data. The intuition is that these Universum data contain information about the problem domain of interest and this information can be used to train an improved boosting classifier.

### 2.1 Motivation

Weston *et al.* [2] presented the algorithm $\mathfrak{U}$SVM, which uses the $\epsilon$-insensitive loss for Universum:

$$\frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{M}\varphi[y_i f_{\boldsymbol{w},b}(\boldsymbol{x}_i)] + D\sum_{j=1}^{N}\rho[f_{\boldsymbol{w},b}(\boldsymbol{x}'_j)], \quad (1)$$

where $\varphi_\theta[t] = \max\{0, \theta - t\}$ is the hinge loss function, and $\rho[t] = \varphi_{-a}[t] + \varphi_{-a}[-t]$ is the $\epsilon$-insensitive loss (see Figure 1). $f_{\boldsymbol{w},b}(\boldsymbol{x}) = \boldsymbol{w}^\top\Phi(\boldsymbol{x}) + b$ is the learned classifier. $\Phi(\cdot)$ is the feature mapping function, which may only be available through its inner product. Instead of training an SVM with the Universum, we are interested in designing a boosting algorithm with Universum.

Collins *et al.* [17] showed that AdaBoost is equivalent to minimize the exponential loss with regularization. Shen and Li presented an $\ell_1$-norm regularized version of the standard AdaBoost, named AdaBoost-CG [10], which can be expressed as:

$$\min_{\boldsymbol{w}} \frac{1}{M}\sum_{i=1}^{M}\exp(-y_i F(\boldsymbol{x}_i)) + D\boldsymbol{1}^\top\boldsymbol{w}, \ \text{s.t.}\ \boldsymbol{w} \geq \boldsymbol{0}, \quad (2)$$

where $D$ controls the trade-off between the exponential loss and the regularization term. Of course other regularization functions may be used here.

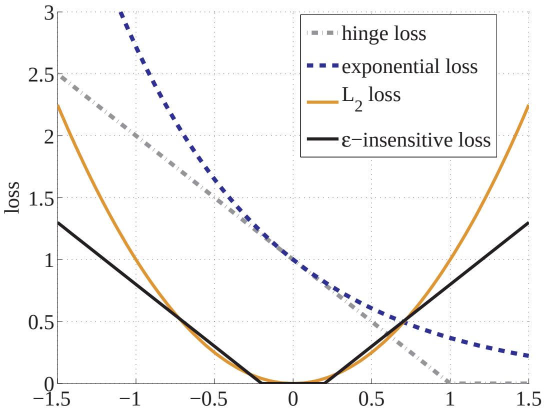


Fig. 1: Four loss functions mentioned in this work. The hinge loss and the exponential loss decrease monotonously with the margin and can be applied on labeled data. The $\varepsilon$-insensitive loss and the $\ell_2$ loss penalize decision values with large absolute values.

Our goal is to find an optimal $\boldsymbol{w}$, which maximizes margins for labeled data and simultaneously minimizes the absolute values of margins for Universum data. The second loss plays the role of maximizing the observed contradictions, which helps to control the generalization capacity of the learned machine as shown in [1]. In this work, we mainly use the $\ell_2$ loss for the Universum data for its simplicity. Note that we can also use other loss functions such as the $\varepsilon$-insensitive loss.

Similar to AdaBoost-CG, we add an $\ell_2$ loss for the Universum data into the optimization problem (2), which implements $\mathfrak{U}$Boost:

$$\min_{\boldsymbol{w}} \frac{1}{M}\sum_{i=1}^{M}\exp(-y_i F(\boldsymbol{x}_i)) + \frac{C}{2N}\sum_{j=1}^{N}F(\boldsymbol{x}'_j)^2 + D\boldsymbol{1}^\top\boldsymbol{w},$$
$$\text{s.t.}\ \boldsymbol{w} \geq \boldsymbol{0}. \quad (3)$$

Here $F(\boldsymbol{x}_i) = H_i\boldsymbol{w}$ is the learned strong classifier.

Figure 1 illustrates the loss functions that can be used in $\mathfrak{U}$SVM and $\mathfrak{U}$Boost. The hinge loss and the exponential loss can be applied on the margins of labeled data (*i.e.*, $y_i F(\boldsymbol{x}_i)$, $i = 1, 2, \dots, M$), while the $\epsilon$-insensitive loss or the $\ell_2$ loss can be applied on the decision values of the Universum data (*i.e.*, $F(\boldsymbol{x}'_j)$, $j = 1, 2, \dots, N$). Intuitively, when the $\epsilon$-insensitive loss or the $\ell_2$ loss is applied to the Universum data, it encourages the decision values on the Universum data to stay close to the decision hyper-plane.

It is difficult to *directly* optimize the problem (3) because we do not have access to all the weak classifiers in most cases. In other words, the matrix $H$ is unknown. Even if we do know all the weak classifiers, usually the number of all possible weak classifiers is extremely large, which corresponds to the variable $\boldsymbol{w}$ with exponentially large dimensions. Next, we derive a meaningful Lagrange dual problem of (3) and show how to use column generation to approximately solve (3) based on the derived dual.

### 2.2 The Lagrange dual of $\mathfrak{U}$Boost

First, we introduce two auxiliary variables $\boldsymbol{z} \in \mathbb{R}^M$ and $\boldsymbol{z}' \in \mathbb{R}^N$ and rewrite (3), where $z_i = -y_i H_i\boldsymbol{w}$ and $z'_j = H'_j\boldsymbol{w}$:

$$\min_{\boldsymbol{w}} \frac{1}{M}\sum_{i=1}^{M}\exp(z_i) + \frac{C}{2N}\sum_{j=1}^{N}{z'_j}^2 + D\boldsymbol{1}^\top\boldsymbol{w},$$
$$\text{s.t.}\ z_i = -y_i H_i\boldsymbol{w}, \forall i;\ z'_j = H'_j\boldsymbol{w}, \forall j;\ \boldsymbol{w} \geq \boldsymbol{0}. \quad (4)$$

It is these two sets of auxiliary variables that lead to an interesting dual problem as we show next. Note that given a primal optimization problem, one can have many different forms of dual problems—not all of these dual problems can lead to a column generation based optimization strategy.

The Lagrangian $L(\cdot)$ associated with the problem (4) is

$$L(\underbrace{\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{z}'}_{\text{primal}}, \underbrace{\boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{v}}_{\text{dual}}) = \frac{1}{M}\sum_{i=1}^{M} \exp z_i + \frac{C}{2N}\sum_{j=1}^{N} {z'_j}^2 + D\mathbf{1}^\top \boldsymbol{w}$$
$$- \boldsymbol{\lambda}^\top \boldsymbol{w} - \sum_{i=1}^{M} u_i(z_i + y_i H_i \boldsymbol{w}) - \sum_{j=1}^{N} v_j(z'_j - H'_j \boldsymbol{w}). \quad (5)$$

Then the dual function is

$$g(\boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{v}) = \inf_{\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{z}'} L(\boldsymbol{w}, \boldsymbol{z}, \boldsymbol{z}', \boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{v}) =$$
$$-\sup_{\boldsymbol{z}} \left( \boldsymbol{u}^\top \boldsymbol{z} - \frac{1}{M}\sum_{i=1}^{M} \exp z_i \right) - \sup_{\boldsymbol{z}'} \left( \boldsymbol{v}^\top \boldsymbol{z}' - \frac{C}{2N}\sum_{j=1}^{N} {z'_j}^2 \right)$$
$$+ \inf_{\boldsymbol{w}} \overbrace{\left( D\mathbf{1}^\top - \boldsymbol{\lambda}^\top - \sum_{i=1}^{M} u_i y_i H_i + \sum_{j=1}^{N} v_j H'_j \right)}^{\text{must be } \mathbf{0}} \boldsymbol{w},$$
$$= -\sum_{i=1}^{M} u_i(\log(M u_i) - 1) - \frac{C}{2N}\sum_{j=1}^{N} v_j^2. \quad (6)$$

By collecting all the constraints from the Lagrangian equation and eliminating $\boldsymbol{\lambda}$, we obtain the dual problem of (4) as follows:

$$\begin{aligned} \max_{\boldsymbol{u}, \boldsymbol{v}} \quad & -\sum_{i=1}^{M} u_i(\log(M u_i) - 1) - \frac{C}{2N}\sum_{j=1}^{N} v_j^2 \\ \text{s.t.} \quad & \sum_{i=1}^{M} u_i y_i H_i - \sum_{j=1}^{N} v_j H'_j \le D\mathbf{1}^\top. \end{aligned} \quad (7)$$

Since the primal problem (4) is convex and strictly feasible, and the Slater's condition holds, strong duality holds between problems (4) and (7). This guarantees that the optimal objective value of (4) is equal to the optimal objective value of (7).

Based on the KKT optimality conditions [8], the gradient of Lagrangian over primal and dual variables must vanish at the optimal point. We can establish the relationship between the optimal primal variables and the optimal dual variables:

$$u_i^* = \frac{1}{M}\exp z_i^*, \quad v_j^* = \frac{C}{N} z_j'^*. \quad (8)$$

These equalities enable us to compute the optimal dual variables from the primal variables.

## 2.3 Optimization of $\mathfrak{U}$Boost using column generation

Since the total number of possible weak classifiers is generally very large (even infinite), we have difficulties to directly solve the problem (4) or its dual (7). To tackle this difficulty, an optimization technique called column generation can be applied to the dual problem. As an iterative method, column generation adds the most violated constraint to the restricted master problem at each iteration. Therefore, at each iteration we solve a relaxed version of the original problem.

For $\mathfrak{U}$Boost, each constraint in the dual problem corresponds to a selected weak classifier in the primal. In theory, any violated constraint can be added into the master problem in the iterative procedure of column generation. However, in practice, to speed up the convergence of column generation, we use the following criterion to find the most violated constraint to add into the master problem, namely the best weak classifier at each round:

$$h^\star(\cdot) = \underset{h(\cdot)}{\operatorname{argmax}} \left( \sum_{i=1}^{M} u_i y_i h(\boldsymbol{x}_i) - \sum_{j=1}^{N} v_j h(\boldsymbol{x}'_j) \right). \quad (9)$$

**Algorithm 1** Column generation for $\mathfrak{U}$Boost.

```
Input:
  • A set of labeled examples $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_M, y_M)\}$, and a set
    of unlabeled Universum examples $\{\boldsymbol{x}'_1, \dots, \boldsymbol{x}'_N\}$.
  • Termination threshold $\varepsilon > 0$;
  • Regularization parameter $C$ and $D$;
  • Maximum iteration $k_{\max}$ (optional) .
Initialization:
  1) $k = 0$ (no weak classifiers selected);
  2) $\boldsymbol{w} = \mathbf{0}$ (all primal coefficients are zeros);
  3) $u_i = \frac{1}{M}$, $i = 1, \dots, M$; $v_j = \frac{1}{N}$, $j = 1, \dots, N$.
while true do
    1) Find a new weak classifier $h^\star(\cdot)$ by solving (9);
    2) Check the termination condition:
       if $\sum_{i=1}^{M} u_i y_i h(\boldsymbol{x}_i) - \sum_{j=1}^{N} v_j h(\boldsymbol{x}'_j) < D + \varepsilon$,
       then break;
    3) Add $h^\star(\cdot)$ to the restricted master problem, which
       corresponds to a new constraint in the dual;
    4) Solve the dual problem (7) or the primal problem (4) to
       obtain updated $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$.
    5) Increment weak classifiers $k = k + 1$ ;
    6) (optional) if $k \ge k_{\max}$, then break.
Output: The final classifier is $F(\boldsymbol{x}) = \sum_k w_k h_k(\boldsymbol{x})$.
```

Algorithm 1 summarizes the framework of column generation for $\mathfrak{U}$Boost. At each iteration, we can solve the primal problem or the dual problem, which are equivalent. In our case, the primal problem (3) is a smoothed convex minimization problem with simple non-negativeness constraints, and the dual problem is a complicated nonlinear concave maximization problem with linear constraints. Usually interior point methods are used to solve problems like (7) [8]. So in practice, we employ L-BFGS-B [18] to solve the primal problem (3), which is much faster than to solve the dual problem.

L-BFGS-B, which is a quasi-Newton algorithm, can be used to optimize a bound-constrained convex problem. This tool is efficient and use less memory to store the value and gradient of the objective function. Since the optimal points for adjacent iterations should not be far away, the results of the last iteration is used as a "warm-start" initialization point for the current iteration. In our experiments, this warm-start dramatically reduces the computation time.

We can see that the primal variable $\boldsymbol{w}$ is the weak classifier weights, while $\boldsymbol{z}$ stands for the margins of labeled examples and $\boldsymbol{z}'$ stands for the "absolute margins" of unlabeled examples. Furthermore, the dual variables $\boldsymbol{u}$ and $\boldsymbol{v}$ are the weights for labeled and unlabeled examples, respectively. Similar to standard boosting algorithms like AdaBoost, the weights measure how important they are for selecting the best weak classifier at each iteration. Different from standard boosting algorithms, the weight of the Universum data $\boldsymbol{v}$ can be negative. So in $\mathfrak{U}$Boost, the magnitude of $v_j$, which shows how much a particular Universum datum deviates from the decision hyper-plane, is the importance measure.

**$\epsilon$-insensitive loss** Here we discuss the case that $\epsilon$-insensitive loss is used on the Universum data. The primal problem can be written as:

$$\begin{aligned}
\min_{\boldsymbol{w},\boldsymbol{\eta},\xi} \; & \frac{1}{M}\sum_{i=1}^{M}\exp(z_i) + \frac{C}{2N}\sum_{j=1}^{N}(\eta_j+\xi_j) + D\mathbf{1}^\top \boldsymbol{w}, \\
\text{s.t. } & z_i = -y_i H_i \boldsymbol{w}, \forall i; \qquad (10)\\
& H'_j \boldsymbol{w} \le \epsilon + \eta_j; H'_j\boldsymbol{w} \ge -\epsilon - \xi_j, \forall j; \\
& \boldsymbol{\eta} \ge \mathbf{0}, \boldsymbol{\xi} \ge \mathbf{0}, \boldsymbol{w} \ge \mathbf{0}.
\end{aligned}$$

The dual problem is

$$\begin{aligned}
\max_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{s}} \; & -\sum_{i=1}^{M} u_i(\log(Mu_i)-1) - \epsilon\mathbf{1}^\top(\boldsymbol{v}+\boldsymbol{s}) \\
\text{s.t. } & \sum_{i=1}^{M} u_i y_i H_i + \sum_{j=1}^{N}(s_j - v_j)H'_j \le D\mathbf{1}^\top. \qquad (11)\\
& \mathbf{0} \le \boldsymbol{v} \le \frac{C}{2N}; \mathbf{0} \le \boldsymbol{s} \le \frac{C}{2N}.
\end{aligned}$$

The column generation technique can be applied as before. The optimization procedure including the subproblem of finding the most violated constraint is similar to the case of the square loss. However, now the primal is not a simple convex optimization anymore—it has a set of linear constraints. So L-BFGS-B is not applicable. In general, the $\epsilon$-insensitive loss is more expensive to solve mainly because it is not smooth. In this case, we can use interior point methods based solvers such as Mosek[1] to solve the primal problem (or the dual problem), which outputs both the primal and dual solutions. Therefore, in our experiments, we have used the square loss for simplicity.

## 3 Experiments

We compare the proposed $\mathfrak{U}$Boost algorithm against a few existing boosting algorithms on various data sets in this section.

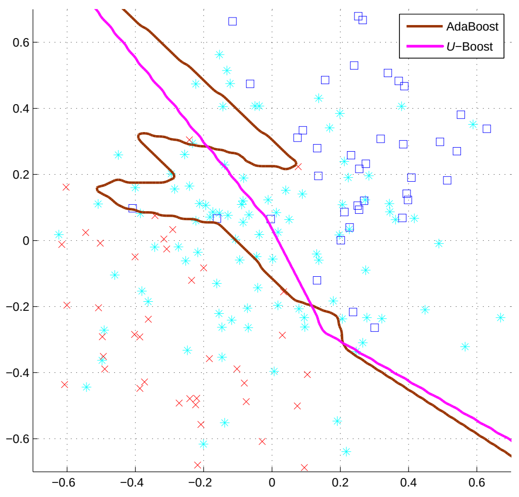
### 3.1 Artificial data

We construct two sets of 2-dimensional data to show the difference between AdaBoost and $\mathfrak{U}$Boost, intuitively. The first data set contains 100 labeled data (50 positive data and 50 negative data) and 100 Universum data. The labeled data follow the Gaussian distribution with the mean of $\mu^{\pm}_{1,2} = \pm 0.3$ and the standard deviation of $\sigma_{1,2} = 0.08$. The unlabeled data follow the Gaussian distribution with the mean of $\mu_{1,2} = 0$ and the standard deviation of $\sigma_{1,2} = 0.1$. 25 weighted FDA weak classifiers are learned by AdaBoost and $\mathfrak{U}$Boost, respectively.

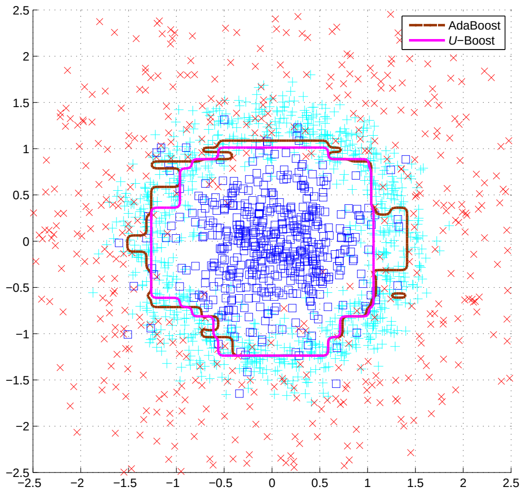
The second data set contains 1000 labeled data (500 positive data; 500 negative data) and 1000 Universum data. The positive data follows Gaussian distribution ($\mathcal{N}(0, 0.25\mathbf{I})$), and the negative data form a circle with radius 2.0. The Universum data form a circle with radius 1.3. 200 decision stumps are learned by AdaBoost and $\mathfrak{U}$Boost.

From Figure 2 (a) and (b), we can find that, in both cases, when the same type and number of weak classifiers are used, the decision boundaries of AdaBoost seems to be more complex than $\mathfrak{U}$Boost. AdaBoost tends to correctly classify all the labeled examples, resulting in an over-fitting decision boundary. However, $\mathfrak{U}$Boost try to make a trade-off between the classification accuracy on labeled data and the contradictions on the unlabeled data, which prevents it from overfitting to outliers.

1. http://www.mosek.com



(a) Artificial data (two Gaussians)



(b) Artificial data (circle and pie)

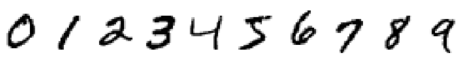
Fig. 2: Decision boundaries learned by AdaBoost and $\mathfrak{U}$Boost on artificial data. Plot (a) shows the result to classify two Gaussian distributed data, with Gaussian distributed Universum data in the middle of positive and negative data points. Plot (b) shows the result to classify a circle and a pie, with the Universum data forming a circle between two classes.

### 3.2 Handwritten symbols

In this section, we compare the performances of $\mathfrak{U}$Boost and a few other classifiers including AdaBoost, AdaBoost-CG [10], LPBoost [15] on the task of handwritten symbols classification. Three data sets are evaluated: MNIST[2], USPS[3] and ABCDETC [2]. The advantage of using handwritten symbols datasets is that, it is easy to obtain a Universum set. For example, if we use digits "3" and "6" as the labeled data, then all the other digits can be used as Universum data. Figure 3 shows some samples from these three data sets.

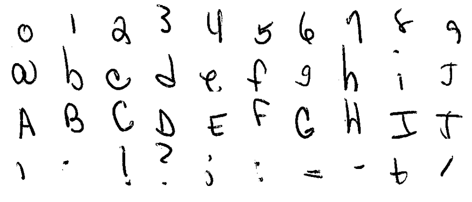
Our experiments are carried out on raw pixel features as

2. http://yann.lecun.com/exdb/mnist/
3. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/



(a) MNIST

(b) USPS



(c) ABCDETC

Fig. 3: Samples from the data sets of MNIST, USPS and ABCDETC.

well as pyramid histogram of gradient (PHOG) features[4]. The raw pixel features are straightforward but have relatively weaker performance in most cases. The PHOG feature is a popular descriptor in computer vision community and shown to achieve better performance for digit classification [19].

First, normalization is performed on images such that the $\ell_2$-norm of raw pixel values is 1. Second, given an input image, the local gradients of pixels are computed by performing convolution on the image with oriented derivative filters. For each pixel, there are two responses in the horizontal and vertical directions, based on which the magnitude and orientation are calculated. Third, histograms can be constructed for cells with a pyramid of sizes and half-size overlap. In each cell, the magnitude of all pixels are aggregated into a set of orientation bins by linear interpolation between bin centers to avoid aliasing. The final feature space is concatenated by all histograms, which are weighted corresponding to cell sizes.

There are a few options to generate gradient histograms, such as types of filters, signed or unsigned orientation angles and number of orientation bins. We make the choices which are reported as the best in [19], namely using the oriented Gaussian derivative (OGF) filter, signed orientation angle (0-360) and 12 orientation bins.

The number of weak classifiers for AdaBoost is cross validated from $\{100, 200, 500, 1000\}$. The regularization parameter of LPBoost (the parameter $D$ in Equation 4 of [15]) is chosen from $\{0.001,\ 0.005,\ 0.01,\ 0.02,\ 0.05,\ 0.1,\ 0.2, 0.5\}$ using cross validation. There are two parameters $C$ and $D$ in our universum boosting $\mathfrak{U}$Boost. Both of these two parameter are chosen from candidates $\{2^{-17},\ 2^{-15},\ 2^{-13},\ 2^{-11},\ 2^{-9},\ 2^{-7},\ 2^{-5}\}$ with cross validation. The pair of parameters with the highest accuracy on validation set are selected. The $\mathfrak{U}$Boost algorithm is terminated when the stopping criterion is met or the maximum number of weak classifiers is reached (we set the maximum number of weak classifiers to 1000 in the experiments). All the results reported in this section are the average of 10 independent runs. We have used these setting for all the experiments in this work. We use decision stumps as weak classifiers. Decision stump is a single-level decision tree on single feature dimension, which is the simplest decision tree.

4. The code are downloaded from the authors' website http://www.cs.berkeley.edu/~smaji/projects/digits/

**MNIST** The MNIST data set has a set of $28 \times 28$ gray-level images of handwritten digits (0-9). There are 60000 examples (about 6000 per digit) for training and 10000 examples (about 1000 per digit) for testing.

We construct gradient histograms with cell sizes $14 \times 14$, $7 \times 7$, $4 \times 4$ and weights 1, 2, 4, which span a 2172 dimensional feature space.

Like the experiments performed in [2], we use digits "5" and "8" to form a binary classification problem. We randomly split "5"s and "8"s in the original training set (totally 11272 examples) into the training subset and validation subset with various sizes (500, 1000, 2000 or 3000 for training and the 1000 for validation), and still use "5"s and "8"s of the original test set for testing performance. The test set size is 1866.

Weston *et al.* [2] indicated that digits "3" and "6" are the most helpful as Universum data to improve classification performance. Therefore, we use all "3"s and "6"s in the original training set (totally 12049 examples) as Universum data.

The results are demonstrated in Table 1. $\mathfrak{U}$Boost beats AdaBoost, AdaBoost-CG [10] and LPBoost [15] in all cases with different features and training sizes. We have compared our algorithm with two totally corrective boosting methods, AdaBoost-CG and LPBoost, to justify whether the improvement is brought by the totally corrective optimization. Actually on this data set, AdaBoost-CG and LPBoost perform slightly worse than AdaBoost. It confirms the usefulness of Universum data. More experiments in the sequel affirm this observation. As an illustrative comparison, RBF kernel $\mathfrak{U}$SVM's results reported in [2] are 1.60%, 1.10%, 0.75%, 0.55% for training size $500, 1000, 2000, 3000$ respectively. So our $\mathfrak{U}$Boost with raw pixels is worse than these results but $\mathfrak{U}$Boost with PHOG performs better than RBF $\mathfrak{U}$SVM.

We run one more experiment to compare boosting with $\mathfrak{U}$SVM. See Table 2 for details. For this experiment, the Universum data are generated by random averaging of the positive and negative data. Random averaging means that we create an artificial image by randomly selecting a pair of training data points, one for each class; and then constructing the mean of these two data points. We have reported the results of $\mathfrak{U}$SVM with both linear and RBF kernel. $\mathfrak{U}$Boost performs better than linear $\mathfrak{U}$SVM but worse than nonlinear $\mathfrak{U}$SVM, which is expected. We again observe that $\mathfrak{U}$Boost performs better than conventional boosting.

**USPS** The USPS dataset contains 7291 examples for training and 2007 for test, in which each example is a $16 \times 16$ graylevel image of one digit.

Likewise with MNIST, we run experiments to classify "5" and "8" with "3" and "6" as Universum data. In this setup, there are 1098 labeled training examples, 326 labeled test examples and 1322 Universum examples.

The cell sizes for gradient histograms are $8 \times 8$, $4 \times 4$ and $2 \times 2$, and the weights with respected to these sizes are 1, 2 and 4. Finally, 2688-dimensional features are generated.

First, we randomly sample 100, 300, or 600 examples from the original training set for training and 400 examples are for validation. All the 1322 examples of digits "3" and "6" are used as Universum data. Both raw pixel feature and PHOG feature are evaluated individually. Table 3 shows the results for this setup, and in most cases, our algorithm achieves better performance over AdaBoost, AdaBoost-CG and LPBoost. Especially on the PHOG feature, when the training set size is

TABLE 1: Classification error rates (standard deviations) in percentage on the MNIST data. Digits 5 and 8 are used as labeled data, and digits 3 and 6 as Universum data. The validation size is 1000.

| Image feature | Method | Training data size | | | |
|---|---|---|---|---|---|
| | | 500 | 1000 | 2000 | 3000 |
| Raw pixels | AdaBoost | 5.58 (0.48) | 4.57 (0.46) | 3.88 (0.41) | 3.26 (0.41) |
| | AdaBoost-CG | 5.84 (0.53) | 4.68 (0.36) | 3.87 (0.27) | 3.94 (0.45) |
| | LPBoost | 6.61 (1.02) | 7.80 (3.81) | 6.51 (0.82 | 6.31 (0.84) |
| | 𝔘Boost | 5.39 (0.66) | 4.36 (0.34) | 3.31 (0.29) | 3.10 (0.21) |
| PHOG | AdaBoost | 0.46 (0.12) | 0.45 (0.10) | 0.27 (0.06) | 0.14 (0.05) |
| | AdaBoost-CG | 0.67 (0.20) | 0.35 (0.14) | 0.30 (0.10) | 0.30 (0.11) |
| | LPBoost | 0.92 (0.24) | 0.71 (0.59) | 0.66 (1.80) | 0.42 (0.24) |
| | 𝔘Boost | 0.28 (0.09) | 0.24 (0.08) | 0.24 (0.13) | 0.13 (0.07) |

TABLE 2: Comparison of 𝔘Boost and 𝔘SVM on the MNIST data set using raw pixels. Samples of handwritten digits 5 and 8 are used for classification. Each sample is represented as a 784 dimensional vector. We set the training set size as 1000, validation set size as 1000 and test set size as 1866. 1000 Universum samples are generated via random averaging.

| Method | |
|---|---|
| AdaBoost | 4.57 (0.46) |
| AdaBoost-CG | 5.01 (0.25) |
| LPBoost | 5.84 (0.47) |
| 𝔘Boost | 4.46 (0.33) |
| 𝔘SVM (Linear) | 4.62 (0.37) |
| 𝔘SVM (RBF) | 1.20 (0.19) |

TABLE 3: Classification error rates (standard deviations) in percentage on the USPS data with different training data sizes. The validation size is 400. Digits "5" and "8" construct labeled data, and the Universum data are made up of digits "3" and "6".

| Feature | Method | Training data size | | |
|---|---|---|---|---|
| | | 100 | 300 | 600 |
| Raw pixels | AdaBoost | 5.82 (0.90) | 4.57 (1.01) | 3.96 (0.59) |
| | AdaBoost-CG | 6.84 (1.05) | 4.88 (0.82) | 4.08 (0.66) |
| | LPBoost | 6.90 (1.07) | 4.91 (1.22) | 4.42 (1.01) |
| | 𝔘Boost | 5.82 (0.43) | 4.51 (1.09) | 3.65 (0.99) |
| PHOG | AdaBoost | 3.22 (1.33) | 1.75 (0.46) | 1.69 (0.39) |
| | AdaBoost-CG | 3.62 (1.26) | 1.90 (0.50) | 1.90 (0.52) |
| | LPBoost | 3.37 (1.26) | 2.09 (0.50) | 2.15 (0.50) |
| | 𝔘Boost | 1.90 (0.40) | 1.10 (0.30) | 1.44 (0.96) |

small, the improvement is significant.

Second, we use the same training data size and different Universum data sizes (100, 300, 500, 1000, 1322). Only the PHOG feature is used for this setup. From Table 4, we can see that, 𝔘Boost outperforms the other three boosting algorithms in all cases. On the other hand, the performance of 𝔘Boost improves with growing Universum data size.

**ABCDETC** The ABCDETC data set used in [2] collects 19646 images of 78 common symbols, including digits ("0-9"), uppercase letters ("A-Z"), lowercase letters ("a-z"), and other symbols (", . ! ? ; : = − + / ( ) \$ % " @"). Those symbols are written in pen by 51 subjects (5 examples per symbol per subject), and then saved as $100 \times 100$ binary images.

The original images are shrunk into $32 \times 32$ graylevel images by bilinear interpolation in nearest $2 \times 2$ neighborhood (anti-aliasing is performed). Gradient histograms are computed with cell sizes $16 \times 16$, $8 \times 8$, $4 \times 4$ and corresponding weights 1, 2, 4, making 2688 feature dimensions.

In this experiment, we try to classify lower-case letters "a" and "b", with various training/validation/test splits (20, 50, 100, 150 or 200 for training, 200 for validation and the rest for test) and Universum data sets (digits, uppercase letters, lowercase letters without "a" and "b", other symbols). Since we use all possible examples to construct Universum data, the four Universum data sets' sizes are not the same: 2544, 6569, 5975 and 4049 respectively.

Table 5 reports the results. Again we can find that, 𝔘Boost outperforms AdaBoost and other boosting in most cases.

Experiments on these three handwritten digits recognition tasks clearly show the effectiveness of the proposed 𝔘Boost algorithm. We can draw the conclusion that properly designed Universum data indeed help to improve the classification accuracy in most cases. This finding is consistent with the results in 𝔘SVM [2].

## 3.3 Gender classification using face images

It has been shown in [7] that Universum data help in the context of gender classification using SVM. Here we follow the experiment protocol of [7] to verify if Universum data improve the performance of 𝔘Boost for gender classification. As in [7], we collect face images of 32 male and 20 female, 10 face images per person[5]. The experiments are conducted on the original data without any normalization or histogram equalization. Each images is converted to 256-level gray-scale and down-sampled to $45 \times 50$ pixels to form a 2250 dimensional vector. See [7] for details about this data set.

We evaluate the performance of our method on this data set with three different settings. First, we adopt the experiment setup of [7], in which 13 subjects are randomly selected for training, and the rest 39 subjects for testing. For each individual, three face images are randomly selected—one for training, one for cross validation and the third one for testing. Therefore, the size of the training and validation set is 13 and test size is 39. Second, For each individual, three images are randomly selected for training, validation and testing. So the size of training, validation and testing set are all 52. Third, two images are randomly selected from each individual. So the size of training, validation and testing set are all 104. The experiment results are summarized in Table 6. The mean and standard deviation are reported on 10 independent runs. Universum samples are generated from the training samples by random averaging of pairs of male and female face images, as shown in Figure 4. Clearly, our 𝔘Boost outperforms AdaBoost, AdaBoost-CG and LPBoost in most cases, especially when the training size is small. Again, we see that 𝔘Boost performs better than the two totally corrective boosting algorithms, too. As a comparison, the best error rate of 𝔘SVM in [7] is $10.8\% \pm 2.4\%$ using 13 training examples. Note that the selected face subjects in their work can differ from ours.

## 3.4 Action recognition

In this section we test our algorithm on the KTH human action recognition data set [20]. The KTH data set consists of 2387 video sequences. They can be categorized into six types of

5. http://cswww.essex.ac.uk/mv/allfaces/faces94.html

TABLE 4: Classification error rates (standard deviations) in percentage on the USPS data set using PHOG features with different Universum data sizes. The training size is 500 and validation size 400. Digits "5" and "8" are the labeled data; the Universum data are digits "3" and "6". For AdaBoost, AdaBoost-CG and LPBoost, the results do not change because no Universum data are used.

| Method | Universum data size | | | | |
|---|---|---|---|---|---|
| | 100 | 300 | 500 | 1000 | 1322 |
| AdaBoost | 1.87 (0.60) | 1.87 (0.60) | 1.87 (0.60) | 1.87 (0.60) | 1.87 (0.60) |
| AdaBoost-CG | 1.93 (0.65) | 1.93 (0.65) | 1.93 (0.65) | 1.93 (0.65) | 1.93 (0.65) |
| LPBoost | 1.93 (0.56) | 1.93 (0.56) | 1.93 (0.56) | 1.93 (0.56) | 1.93 (0.56) |
| $\mathfrak{U}$Boost | 1.66 (0.58) | 1.53 (0.20) | 1.44 (0.63) | 1.289 (0.35) | 1.17 (0.52) |

TABLE 5: Classification errors (standard deviations) in percentage on ABCDETC dataset. Lowercase letters "a" and "b" are used as labeled data. Four types of Universum data are evaluated, which are digits (0-9), uppercase letters ($A$-$Z$), lowercase letters ($c$-$z$), and other symbols (, . ! ? ; : = − + / ( ) \$ % " @ ).

| Feature | Method | Type of Universum data | Training data size | | | | |
|---|---|---|---|---|---|---|---|
| | | | 20 | 50 | 100 | 150 | 200 |
| Raw pixels | AdaBoost | None | 44.53 (6.49) | 38.96 (5.4) | 31.87 (2.84) | 30.63 (3.83) | 27.61 (3.68) |
| | AdaBoost-CG | None | 45.29 (5.72) | 39.38 (5.25) | 34.16 (3.09) | 33.65 (3.96) | 29.36 (5.54) |
| | LPBoost | None | 43.81 (5.46) | 38.46 (5.07) | 32.39 (1.60) | 31.00 (4.51) | 29.17 (4.85) |
| | $\mathfrak{U}$Boost | Lowercase | 39.83 (2.63) | 35.10 (4.61) | 30.24 (2.42) | 28.99 (3.80) | 26.42 (3.37) |
| | | Uppercase | 41.59 (2.79) | 37.68 (2.66) | 30.81 (3.50) | 28.11 (3.64) | 28.53 (4.46) |
| | | Digits | 39.72 (2.65) | 35.79 (4.01) | 30.00 (2.04) | 29.43 (3.20) | 25.60 (4.07) |
| | | Symbols | 42.25 (2.96) | 36.83 (3.36) | 33.88 (3.38) | 29.12 (3.75) | 26.06 (3.62) |
| PHOG | AdaBoost | None | 17.33 (6.24) | 11.78 (2.24) | 7.27 (1.86) | 5.91 (1.68) | 3.85 (1.88) |
| | AdaBoost-CG | None | 18.69 (6.03) | 13.13 (2.69) | 8.71 (2.1) | 7.11 (1.57) | 4.77 (1.82) |
| | LPBoost | None | 17.27 (6.32) | 12.66 (2.37) | 7.89 (2.06) | 6.54 (1.81) | 4.68 (1.86) |
| | $\mathfrak{U}$Boost | Lowercase | 12.35 (2.70) | 8.26 (1.75) | 5.22 (1.45) | 3.77 (1.51) | 2.11 (1.23) |
| | | Uppercase | 13.49 (2.57) | 8.26 (1.86) | 4.35 (0.69) | 3.14 (1.19) | 2.29 (1.74) |
| | | Digits | 12.32 (2.45) | 8.11 (1.98) | 4.88 (1.23) | 3.71 (1.34) | 2.20 (1.38) |
| | | Symbols | 12.73 (2.32) | 7.76 (2.09) | 4.21 (1.46) | 3.46 (1.27) | 2.48 (1.44) |

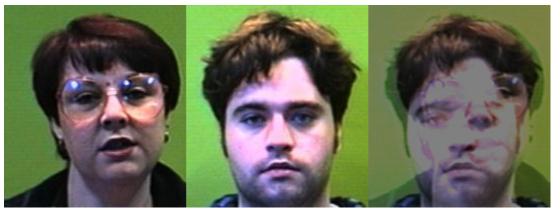


Fig. 4: Examples of female and male faces and the corresponding Universum sample obtained by averaging the two images that have different class labels.

TABLE 6: Classification error rates (standard deviations) in percentage on gender classification with face images. We can see that Universum data improve test accuracy in most cases.

| Method | # Universum | Training data size | | |
|---|---|---|---|---|
| | | 13 | 52 | 104 |
| AdaBoost | – | 26.41 (1.24) | 1.92 (1.57) | 0.96 (0.91) |
| AdaBoost-CG | – | 26.41 (1.24) | 1.92 (1.11) | 1.35 (1.22) |
| LPBoost | – | 28.97 (1.73) | 1.92 (2.22) | 1.32 (1.14) |
| $\mathfrak{U}$Boost | 100 | 15.13 (5.19) | 1.15 (2.07) | 0.67 (0.79) |
| | 500 | 21.03 (6.26) | 0.77 (2.07) | 1.46 (1.52) |
| | 1000 | 15.13 (8.67) | 0.96 (1.36) | 0.58 (0.67) |

human actions including *boxing, hand-clapping, jogging, running, walking and hand-waving*. These actions are performed by 25 subjects and each action is performed multiple times by the same subject. The length of each video is about four seconds at 25 fps, and the resolution of each frame is $160 \times 120$. We randomly split all the video sequences based on the subjects into 10 pairs, each of which contains all the sequences from 16 subjects for training and those from the remaining 9 subjects for testing. The space-time interest points (STIP) [21] are extracted from each video sequence and used to represent the visual content. The descriptors extracted from all the training sequences are clustered into 4000 clusters using $k$-means algorithm. These cluster centers form the visual codebook. Accordingly, each video sequence is characterized by a 4000-dimensional histogram indicating the occurrence of each visual word in this sequence. To achieve a compact and discriminative representation, the visual word merging algorithm, agglomerative information bottleneck (AIB) [22], is applied to merge the histogram bins to reduce the dimensionality. Finally, each video sequence is represented by a $\{50, 100, \text{or } 200\}$-dimensional histogram.

In this experiment, we classify the *hand-clapping* and *hand-waving* sequences as the positive and negative classes, respectively. Out of about 1500 training samples and 850 testing samples, we randomly selected 100 samples for both training set and testing set. The training set are then divided into training subset and validation subset with both 50 samples. We generate about 250 Universum samples for $\mathfrak{U}$Boost from all the training samples by random averaging the extracted features.

We compare different boosting algorithms on three different dimensions (50, 100, 200), and the results of 10 independent runs are reported in Table 7.

As we can see, $\mathfrak{U}$Boost is the best among the compared boosting methods. This is consistent with the previous results on other data sets. We also find that AdaBoost, AdaBoost-CG and LPBoost perform similarly in terms of test accuracy. In [10] it is shown that the totally corrective AdaBoost-CG converges much faster than AdaBoost, but there is no statistically significant difference between AdaBoost-CG and AdaBoost. We observe the same phenomenon.

## 3.5 Traffic sign classification

In this section, we perform traffic sign classification on the German Traffic Sign Recognition Benchmark (GTSRB)[6] data set, which has more than 40 classes and $50,000$ images in total. In our experiment, we select 3 pairs of classes for classification (see examples in Figure 5). Since there is no label information for the on-line test dataset, we generate the training subset, validation subset and test subset from the original training data. Similar to the action classification task, we randomly selected

6. http://benchmark.ini.rub.de/index.php?section=dataset

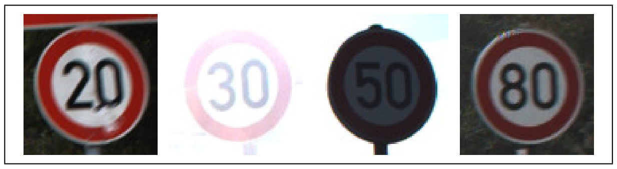


Fig. 5: Samples of traffic signs from the GTSRB data set.

TABLE 7: Classification error rates (standard deviations) in percentage on the KTH data set (training set size and validation set size are 50; test set size is 100). We use the "hand-clapping" and "hand-waving" actions as the positive and negative samples respectively for classification. We have used $250$ Universum samples generated by random averaging.

| Method | Feature dimension | | |
|---|---|---|---|
| | 50 | 100 | 200 |
| AdaBoost | 9.90 (6.13) | 9.10 (6.54) | 5.60 (4.60) |
| AdaBoost-CG | 9.70 (6.77) | 7.40 (5.10) | 5.70 (4.52) |
| LPBoost | 8.60 (5.29) | 7.90 (4.75) | 6.10 (4.53) |
| $\mathfrak{U}$Boost | 6.40 (4.42) | 5.37 (2.92) | 4.90 (4.56) |

100 samples for both training and testing. The training set is then divided into training subset and validation subset with both 50 samples. For both $\mathfrak{U}$Boost and $\mathfrak{U}$SVM, 1000 Universum samples are generated by random averaging. The PHOG features of 1568-dimension are computed for each image. Here we have deliberately used a small amount of training data. When a large training set is used, all the algorithms can achieve very high accuracy and the difference between different methods becomes negligible.

Comparison of different boosting methods and $\mathfrak{U}$SVM with the linear kernel on three pairs of traffic sign sample sets is summarized in Table 8. All the results in this section are the average of 10 independent runs. Here again, $\mathfrak{U}$Boost generally outperforms those boosting methods that do not exploit Universum data. $\mathfrak{U}$Boost is also slightly better than linear $\mathfrak{U}$SVM for two cases out of three.

TABLE 8: Classification error rates (standard deviation) in percentage of different boosting methods and linear $\mathfrak{U}$SVM on three pairs of traffic sign sets.

| Method | "20" vs "30" | "20" vs "50" | "20" vs "80" |
|---|---|---|---|
| AdaBoost | 3.20 (2.90) | 4.60 (4.68) | 3.90 (1.66) |
| AdaBoost-CG | 4.60 (4.90) | 5.70 (4.69) | 7.00 (3.62) |
| LPBoost | 4.60 (4.91) | 5.70 (4.60) | 6.80 (3.91) |
| $\mathfrak{U}$Boost | 2.80 (1.69) | 3.50 (3.06) | 2.60 (1.08) |
| $\mathfrak{U}$SVM | 3.00 (2.58) | 4.70 (1.06) | 1.90 (0.99) |

## 4 Conclusion

We have proposed a new boosting algorithm which can exploit the unlabeled Universum data in the training procedure. We have extended Vapnik's principle of *maximal contradiction on Universum data* to boosting learning. Experiments on a few classification tasks—including handwritten symbol classification, gender recognition, action recognition and traffic sign classification—show promising results over conventional boosting methods such as AdaBoost, AdaBoost-CG and LPBoost, which do not use Universum information. Future work will study on $\mathfrak{U}$Boost with different loss functions and how to effectively design and obtain Universum data for more computer vision and machine learning problems. As pointed out in [2], in some scenarios poorly generated Universum data may not help. It remains unclear about how to generate or select Universum data that always improve the classification accuracy.

## References

[1] V. Vapnik, *Estimation of dependences based on empirical data*. Springer-Verlag, 2006.
[2] J. Weston, R. Collobert, F. Sinz, L. Bottou, and V. Vapnik, "Inference with the Universum," in *Proc. Int. Conf. Mach. Learn.* Pittsburgh, Pennsylvania: ACM, 2006, pp. 1009–1016.
[3] F. H. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf, "An analysis of inference with the Universum," in *Proc. Advances in Neural Information Process. Systems*. Red Hook, NY: MIT Press, 2007, pp. 1369–1376.
[4] D. Zhang, J. Wang, F. Wang, and C. Zhang, "Semi-supervised classification with the Universum," in *Proc. SIAM Int. Conf. Data Mining*. Atlanta, Georgia: SIAM, 2008, pp. 323–333.
[5] B. Peng, G. Qian, and Y. Ma, "View-invariant pose recognition using multilinear analysis and the Universum," in *Proc. Int. Symp. Visual Computing*, vol. 5359, Las Vegas, Nevada, 2008, pp. 581–591.
[6] ——, "Recognizing body poses using multilinear analysis and semi-supervised learning," *Pattern Recogn. Lett.*, vol. 30, no. 14, pp. 1289–1294, 2009.
[7] X. Bai and V. Cherkassky, "Gender classification of human faces using inference through contradictions," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Hong Kong, 2008, pp. 746–750.
[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
[9] R. E. Schapire and Y. Freund, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, no. 3, pp. 297–336, Dec. 1999.
[10] C. Shen and H. Li, "On the dual formulation of boosting algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2216–2231, 2010.
[11] R. Meir and G. Rätsch, *An introduction to boosting and leveraging*, ser. Advanced lectures on machine learning. Springer-Verlag, 2003, pp. 118–183.
[12] C. Shen, P. Wang, and H. Li, "LACBoost and FisherBoost: Optimally building cascade classifiers," in *Proc. Eur. Conf. Comput. Vision*, vol. 2, LNCS 6312, Crete Island, Greece, 2010, pp. 608–621.
[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.
[14] C. Shen and H. Li, "Boosting through optimization of margin distributions," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 659–666, 2010.
[15] A. Demiriz, K. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Mach. Learn.*, vol. 46, no. 1-3, pp. 225–254, 2002.
[16] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent," in *Proc. Advances in Neural Information Process. Systems*. MIT Press, 2000, pp. 512–518.
[17] M. Collins, R. E. Schapire, and Y. Singer, "Logistic regression, AdaBoost and Bregman distances," *Mach. Learn.*, vol. 48, no. 1-3, pp. 253–285, 2002.
[18] C. Zhu, R. H. Byrd, and J. Nocedal, "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997.
[19] S. Maji and J. Malik, "Fast and accurate digit classification," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-159, Nov 2009.
[20] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. Int. Conf. Pattern Recogn.*, vol. 3, 2004, pp. 32–36.
[21] I. Laptev, "On space-time interest points," *Int. J. Comput. Vision*, vol. 64, no. 2-3, pp. 107–123, September 2005.
[22] N. Slonim and N. Tishby, "Agglomerative information bottleneck," in *Proc. Advances in Neural Information Process. Systems*, vol. 12, 1999, pp. 617–623.